UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/674,974	09/30/2003	Yen-Fu Chen	AUS920030588US1	4970

37945 7590 02/11/2011
DUKE W. YEE
YEE AND ASSOCIATES, P.C.
P.O. BOX 802333
DALLAS, TX 75380

EXAMINER	
TIMBLIN, ROBERT M	

ART UNIT	PAPER NUMBER
2167	

NOTIFICATION DATE	DELIVERY MODE
02/11/2011	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

ptonotifs@yeeiplaw.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

Ex parte YEN-FU CHEN, JOHN W. DUNSMOIR,
MARK LOUIS FEINBERG, ABHAY PRADHAN,
and HARI SHANKAR

Appeal 2009-008038
Application 10/674,974
Technology Center 2100

Before JOHN A. JEFFERY, DEBRA K. STEPHENS, and
JAMES R. HUGHES, *Administrative Patent Judges*.

JEFFERY, *Administrative Patent Judge*.

DECISION ON APPEAL[1]

Appellants appeal under 35 U.S.C. § 134(a) from the Examiner's rejection of claims 1, 3-11, 13-20, 22-29, and 32-38. Claims 2, 12, 21, 30, and 31 have been canceled. We have jurisdiction under 35 U.S.C. § 6(b).

---

[1] The two-month time period for filing an appeal or commencing a civil action, as recited in 37 C.F.R. § 1.304, or for filing a request for rehearing, as recited in 37 C.F.R. § 41.52, begins to run from the "MAIL DATE" (paper delivery mode) or the "NOTIFICATION DATE" (electronic delivery mode) shown on the PTOL-90A cover letter attached to this decision.

We affirm-in-part.

STATEMENT OF THE CASE

Appellants invented a schema generation and update program that creates XML schema for a database, copies the database into a hashtable, and compares the database to the hashtable at different intervals. *See generally* Spec. ¶¶ 0001, 0009; Abstract. Claim 1 is reproduced below with a key disputed limitations emphasized:

> 1. A method for validating a plurality of data in a backend driven environment, the method comprising:
>
> creating an XML Schema for a database, wherein the *XML Schema contains a plurality of rules for validating a plurality of data in the database*;
>
> copying the database to a hashtable;
>
> designating a query interval;
>
> upon the occurrence of a query interval, comparing the database to the hashtable;
>
> determining if the database and the hashtable are identical; and
>
> when the database and the hashtable are not identical, creating a new XML Schema;
>
> wherein the creating a new XML Schema includes automatically updating the plurality of rules; and
>
> wherein a new XML Schema is created only when a determination is made that the database and the hashtable are not identical.

The Examiner relies on the following as evidence of unpatentability:

Srivastava	US 2002/0120685 A1	Aug. 29, 2002
Ng	US 6,609,133 B2	Aug. 19, 2003
Janzig	US 7,016,906 B1	Mar. 21, 2006 (filed May 4, 2000)

THE REJECTION

The Examiner rejected claims 1, 3-11, 13-20, 22-29, and 32-38 under 35 U.S.C. § 103(a) as unpatentable over Ng, Janzig, and Srivastava. Ans. 3-17.[2]

THE CONTENTIONS

Regarding representative independent claim 1, the Examiner finds that Ng discloses every recited feature except for (1) creating new XML schema when the database and the hashtable are not identical; (2) designating a query interval; and (3) upon the query interval occurring, comparing the database to the hashtable. (Ans. 3-5). Janzig is cited to teach creating a new schema only when the information is not identical (Ans. 4), and Srivastava is cited to teach using a query interval (Ans. 5).

Appellants argue: (1) neither Ng nor Janzig discloses an XML schema as defined in the disclosure; (2) Ng does not automatically update a schema; (3) Ng, Janzig, and Srivastava fail to teach validating data in the database as defined by the disclosure; (4) Ng only teaches changing a field attribute *may* specify the acceptable value type in a database; (5) Janzig teaches always creating a new schema regardless of the matching outcome; and (6) Srivastava does not teach comparing a database to a hashtable. Br. 10-13.

The issues before us, then, are as follows:

---

[2] Throughout this opinion, we refer to (1) the Appeal Brief filed August 27, 2008 and (2) the Examiner's Answer mailed November 28, 2008.

ISSUES

Under § 103, has the Examiner erred in rejecting claim 1 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested:

(1) an XML schema containing rules for validating data in a database;

(2) comparing the database to the hashtable upon the occurrence of a query interval; and

(3) creating a new XML schema when the database and the hashtable are not identical, where the new XML schema includes automatically updating the rules?

FINDINGS OF FACT (FF)

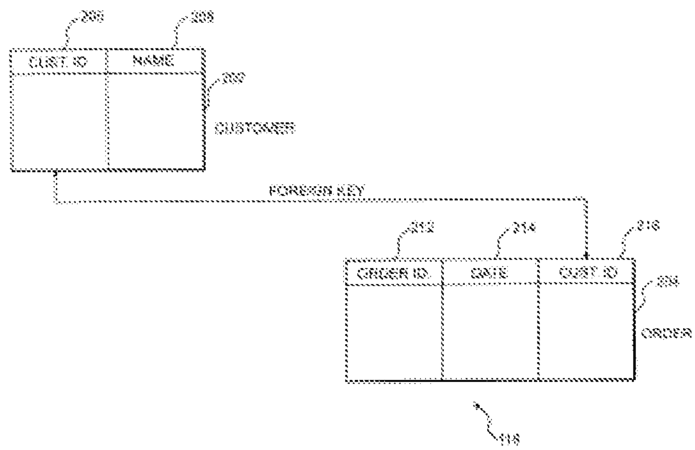
1. Appellants define "XML Schema" as "a computer file containing a plurality of rules for validating data." Spec. ¶ 0024.

2. Appellants define "validate" as "to check data against an XML Schema to determine if the data meets the requirements of the database records, tables, and fields." Spec. ¶ 0023.

3. Ng discloses a data processing system 100 having a computer 101 with memory 104 and secondary storage 106. Memory 104 includes an object-relational mapping tool (ORMT) 114 having an object model 116 and database data structure 115 that reflects a database's (e.g., 118) schema. A schema is logical structure imposed on a relational database that stores data interrelated tables having records and fields. Ng, col. 1, ll. 60-64; col. 4, ll. 53-64; col. 5, ll. 51-52; Figs. 1, 3.

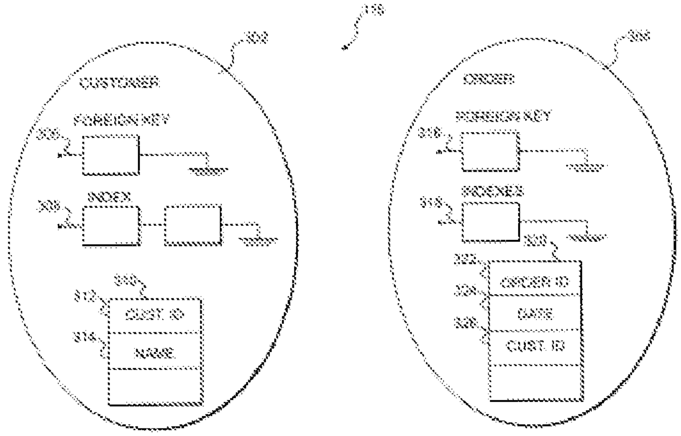
4. Ng's secondary storage 106 contains database 118 and source code 120 having the classes reflecting the schema of database 118. Database 118 contains (1) customer table 202 with customer ID column 206 and name

column 208 and (2) order table 204 with order ID column 212, date column 214, and customer ID column 216. Ng, col. 2, ll. 62-63; col. 4, ll. 61-67; col. 5, ll. 34-50; Figs. 1-2. Ng's Figure 2 is reproduced below:



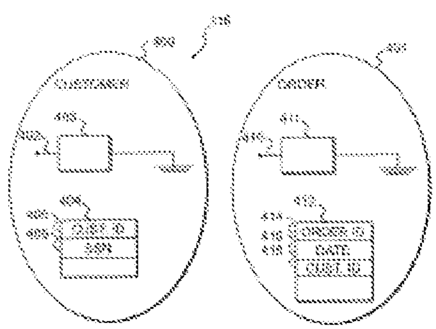
Ng's Figure 2 depicting database 118.

5. The ORMT 114 creates data structure 115 by importing the database's schema, including querying the database 118 to identify and reflect its schema. Objects 302 and 304 having hashtables 310 and 320 are part of data structure 115. Column objects 312, 314 are entries in hashtable 310 and contain data for a particular field. Hashtable 320 has column objects 322, 324, 326 for field or column. Ng, col. 2, ll. 64-65; col. 4, ll. 61-64; col. 5, ll. 51-67; Figs. 1, 3, 6A. Ng's Figure 3 depicting a database data structure is reproduced below:



Ng's Figure 3 depicting a database data structure.

6A. Ng's object model 116 contains the information derived from data structure 115. Ng also discloses object model 116 having object 400 containing hashtable 404 with entries 406 and 408 for each field in the customer table 202. Object 401 contains information for order table 204 and contains hashtable 412 with entries 414, 416, and 418. Ng, col. 2, ll. 66-67; col. 6, ll. 4-29; Fig. 4A. Figure 4A is reproduced below:



Ng's Figure 4A depicting an object model.

6B. Ng discloses that the ORMT 114 allows a programmer to customize the object model 116 having objects (e.g., 400, 401) and represent an intermediate form of the class information before written as source code. Ng states a programmer can: (a) rename a field (e.g., rename field to SSN shown by entry 408 in an object's hashtable 404); (b) change a database

schema (e.g., add a column); or (c) change field attributes (e.g., change the type or whether the field can accept a null value). Ng, col. 2, ll. 66-67; col. 4, ll. 29-51; col. 5, ll. 20-34; col. 6, ll. 1-34; Figs. 1, 2-4A.

7. Ng incorporates changes into the existing object model. This is achieved using the ORMT to create a new database data structure (e.g., shown in Fig. 7) and to compare the original database structure (e.g., Fig. 3) with a newly created one. Noting any differences, the ORMT isolates changes to the schema at step 504 (e.g., addition of address column 716 in hashtable 710), updates the object model at step 506 (e.g., adds address field 802 to object 400 or if any type, name, or number of a field have changes at 1004), and generates new source code (e.g., 804) at step 508. Ng, col. 4, ll. 29-51; col. 7, ll. 1-5, 29-53; col. 8, ll. 10-28; Figs. 5, 7, 8A-B, 10.

8. After database schema changes, the ORMT updates the source code by generating a new source code file (e.g., Fig 4B or 804) from the object model while maintaining customizations. The source code is depicted in a Java programming language. Ng, col. 4, ll. 33-51; col. 5, ll. 20-28; col. 6, ll. 29-34; Figs. 1, 4B, 8B, 12.

9. Srivastava teaches a preference to use XML to validate against an XML schema. The XML service descriptor documents are mapped into a database for access. Srivastava, Abstract; ¶ 0010.

10. Srivastava uses a service engine periodically to update service information and to ensure the system's integrity and proper service operation. The service information is stored in a database. Srivastava, ¶¶ 0068-69, Abstract.

11. Janzig creates a new schema at step 215 from a new description of a database by first examining a time stamp change at step 211. When the time stamp has changed or is different because an administrator reorganized the database, a new schema is created from the database description (e.g., path 212 follows from "YES" at step 211). If the time stamp has not changed (e.g., follows from "NO" at step 211), then the process stops (e.g., "DO NOTHING"). Janzig, col. 9, ll. 13-32; Fig. 19.

ANALYSIS

Claims 1, 10, 20, and 29

1. "Creating an XML Schema and Automatically Updating the Rules in the XML Schema" Limitations

We begin by construing a key disputed limitation of claim 1 which calls for, in pertinent part, an XML schema containing rules for validating data in a database. Appellants define an "XML Schema" as "a computer file containing a plurality of rules for validating data." FF 1. Appellants further define "validate" in terms of the XML schema to mean "to check data against an XML Schema to determine if the data meets the requirements of the database records, tables, and fields." FF 2. Thus, when giving the phrase "the XML schema contains a plurality of rules for validating a plurality of data in the database" in claim 1, its broadest reasonable construction based on Appellants' definitions, a XML schema includes a computer file containing rules for checking data against the computer file to determine if the data meets the database's record, table, and field requirements. *See In re Am. Acad. of Sci. Tech Ctr.*, 367 F.3d 1359, 1364 (Fed. Cir. 2004) (internal citations omitted). Also, because Appellants have

chosen to be their own lexicographer, we are not limited by Ng's use of the word "schema" (*see* FF 3-8), contrary to Appellants' assertions (Br. 10-11). *See Phillips v. AWH Corp.*, 415 F.3d 1303, 1319 (Fed. Cir. 2005) (en banc).

Based on the Appellants' meaning of the term "schema," we agree with the Examiner (Ans. 18-22) that Ng teaches the recited "schema." Ng discloses a computer with memory 104 and secondary storage 106. FF 3. Memory 104 includes an ORMT 114 having a database structure 115 that reflects the database's logical structure by querying the database 118 to identify and import its logical structure. *See* FF 3-5. Additionally, the data structure includes some database data. For example, data structure 115 includes customer ID column object 312, name column object 314, and order ID column object 322 containing information that represents database logical structure and data, including corresponding customer ID column 206, name column 208, and order ID column 212. *See* FF 4-5. In turn, the object model 116 contains objects (e.g., 400, 401) having similar information derived from data structure 115. *See* FF 6A. For example, hashtable 404 contains customer ID entry 406 corresponding customer ID column 206 in the customer table's (e.g., 202) field and also customer ID column object 312 of data structure 115. *See* FF 4-6A.

Ng further explains that the object model can also include or incorporate any programmer customizations and changes to the database's logical structure. *See* FF 6B-7. These include: (a) adding an address column (e.g., 716 in Fig. 7); (b) renaming a field (e.g., rename to SSN as shown by 408 in Fig. 4): and (c) changing a field attribute (e.g., change the type or whether a field can accept a specific value). *See id.* Thus, Ng at least suggests that the programmer can add or rename a field (e.g., rename a field

to SSN), and also change this field's attribute to a specific requirement (e.g., limit the SSN field to accept only nine digit values). *See id.* This change would also, in turn, change the field's attribute by permitted only certain data (e.g., those with nine digits) to be stored in the database. *See* FF 4-7. Thus, as the Examiner explains (Ans. 19), Ng teaches that the object model includes rules (e.g., field attribute defined to accept only certain values, such as nine digits) for checking the data stored in a field to determine if the data meets the database record's requirements or a "schema."

Ng also states these changes are included in the source code file. *See* FF 7-8. As explained above, Ng describes a programmer changing a field attribute (e.g., whether a SSN field can accept a particular value). Thus, when the changes are included the code (e.g., 804), Ng creates yet another computer file containing the rules (e.g., program code that accepts only certain values in the SSN field) for checking field data (e.g., SSN field data) to determine if the data meets the database record requirements (e.g., nine digits). *See* FF 6A-8. That is, Ng's program or source code file checks entries (e.g., entered data) against its computer file to determine if the entry or data meets the specific requirements, and thus, Ng's source code file is a "schema." We therefore find that both Ng's object model and source code map to the recited "schema," as defined by Appellants, in claim 1.

Additionally, Appellants are mistaken that the Examiner relied on inherency when discussing Ng's teaching of changing field attributes. Br. 11. The Examiner explains that Ng teaches — not explicitly discloses — such a feature and allows for field attributes to be changed. *See* Ans. 21. Moreover, since claim 1 is rejected under obviousness, the rejection can properly be based on what a reference teaches or suggests.

Regarding Ng's schema being an "XML" schema, Ng does not explicitly disclose using XML. Nonetheless, Ng teaches that the source code is programmed in Java (FF 8), which supports XML. Moreover, as the Examiner states (Ans. 5, 22), Srivastava teaches mapping XML documents into a database and lends itself to validation. *See* FF 9. Combining Srivastava's teaching of mapping of XML documents to a database with Ng's disclosed database predictably yields no more to an ordinarily skilled artisan than an object model and source code that use XML or an "XML schema" as recited in claim 1. *See KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 416 (2007).

Appellants also assert that Ng does not teach updating a schema, but rather creates a new schema. *See* Br. 10. We disagree. First, we agree with the Examiner that claim 1 recites updating the rules contained in the schema, but not the schema itself. *See* Ans. 20. Second, Ng's changes and customizations are incorporated or updated into the existing object model, and the source code is updated. *See* FF 6B-8. Also, while Ng updates the code by generating new source code (*see* FF 8), generating new code to reflect changes brings the source code up to date or "updates" the schema, given the broadest reasonable construction of the term "update." *See Am. Acad. of Sci.*, 367 F.3d at 1364.

2. *"When the Database and the Hashtable Are Not Identical, Creating a new XML Schema" Limitation*

Appellants further contend that Janzig fails to teach this limitation because the schema is created regardless of the matching outcome. Br. 12. We first note that the rejection is based on the combination of Ng and

Janzig, and attacking references individually does not show nonobviousness where rejections are based on combinations of references. *See In re Merck & Co., Inc.*, 800 F.2d 1091, 1097 (Fed. Cir. 1986). The Examiner states that Ng: (1) determines or compares the database and hashtable, and (2) creates a new schema based on this determination but fails teach creating this schema only when the database and hashtable are not identical. *See* Ans. 4. Appellants have not challenged that Ng fails to teach these features. *See* Br. 10-13. Thus, the § 103 rejection (*see* Ans. 3-5) is only relying on Janzig to teach creating Ng's new schema when the database information has changed (e.g., not identical) but not for the particulars as to the database/hashtable comparison or creating a new schema, as Appellants contend (Br. 12-13).

Second, while Appellants focus on Janzig's later step 217 (Br. 12), Janzig creates or updates a new schema at step 215 from a new description of a database by preliminarily examining a time stamp change at step 211 (FF 11). As the Examiner indicates (Ans. 4, 23), Janzig teaches when the time stamp have changed or are different, the changed database information is updated in a schema (e.g., path 212 follows from "YES" at step 211). *See id.* If the time stamp has not changed (e.g., follows from "NO" at step 211), then Janzig does not update the schema (e.g., "DO NOTHING"). *See id.* Janzig therefore teaches creating a new data structure only when database information has changed. *See id.* Moreover, when combining this teaching with Ng's undisputed database/hashtable comparison, the combined Ng/Janzig/Srivastava method predictably yields no more than creating a new XML schema when the database and the hashtable are not identical. *See KSR*, 550 U.S. at 416. Also, accounting for creative steps and employing inferences reasonably drawn from the cited references, an ordinarily skilled

artisan would have recognized combining Janzig's teaching with Ng to reduce processing demands and only create a new schema when the database information has changed. *See id*. at 418.

3. *"Upon the Occurrence of Query Interval, Comparing the Database to the Hashtable" Limitation*

Appellants also assert that Srivastava fails to teach comparing the database to the hashtable upon the occurrence of a query interval. Br. 13. Again, we note that the rejection is based on the combination of references, and attacking Srivastava individually does not show nonobviousness. *See Merck & Co., Inc.*, 800 F.2d at 1097. The Examiner states that Ng determines or compares the database and hashtable (Ans. 4), and Appellants have not challenged that Ng does not teach this feature (*see* Br. 10-13). The Examiner, however, states Ng fails to teach comparing this information when a query interval occurs. *See* Ans. 5. Thus, the § 103 rejection is only relying on Srivastava to teach designating a query interval and performing services when a query interval occurs. Ans. 5.

Srivastava teaches periodically updating service information stored in a database. FF 10. Periodically updating information, at the very least, suggests to an ordinarily skilled artisan that a query interval has been established to perform the updates at a given period. Thus, when combining such a teaching with Ng's undisputed database/hashtable comparison, the combined Ng/Srivastava method predictably yields no more than comparing the database and hashtable upon a query interval occurring. *See KSR*, 550 U.S. at 416. Moreover, accounting for an ordinarily skilled artisan's

creativity and inferences, an ordinarily skilled artisan would have employed Srivastava's teaching (FF 10) with Ng to ensure that the database and hashtable information are current. *See KSR,* 550 U.S. at 418.

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of: independent claim 1 and claims 10, 20, and 29 not separately argued with particularity (Br. 10-13).

Claim 3, 13, 22, and 32

Regarding representative claim 3, the Examiner relies on Srivastava's teaching of periodically updating information to teach resetting the query interval. Ans. 5. Appellants argue setting the frequency of updates is not the same as resetting the query interval. Br. 14. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 3 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested resetting the query interval?

ANALYSIS

Based on the record before us, we find no error in the Examiner's obviousness rejection of representative claim 3 which calls for, in pertinent part, resetting the query interval. As stated above, Srivastava teaches periodically updating information stored in a database. FF 10. This suggests a period or query interval must be designated to perform the periodic update. *See id.* Also, to designate such a query interval (*see id.*),

Srivastava suggests that period will need to be reset when one time interval (e.g., a period) has been completed and a new one commences. Additionally, one skilled in the art employing reasonable inferences drawn from the cited references and using creative steps would have recognized from Srivastava's teachings to reset a period when one periodic time interval ends. *See KSR*, 550 U.S. at 418. Thus, while we agree with Appellants (Br. 14) that Srivastava's teaching does not explicitly state resetting the query interval, Srivastava nonetheless at least suggests resetting the interval to conduct periodic updates.

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of: claim 3 and claims 13, 22, and 32 not separately argued with particularity (Br. 14).

Claims 4, 14, 23, and 33

Regarding claim 4, the Examiner relies on Ng to teach deleting the hashtable and saving the database as a new hashtable. Ans. 6. Appellants assert that Ng is silent about deleting a hashtable but only deletes entries. Br. 14. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 4 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested (1) deleting the hashtable when the database and hashtable are not identical, and (2) saving the database as a new hashtable?

ADDITIONAL FINDINGS OF FACT (FF)

12. Ng discloses the state operations performed by the ORMT to update the object model once the changes made to the schema have been isolated. These states include determining whether: (1) a new table has been added (i.e., state 1102), and if so, creating a new object (i.e., state 1104); (2) a new column has been added (i.e., state 1106), and if so, adding a new entry to the hashtable (i.e., state 1108); (3) a table has been deleted (e.g., state 1114), and if so, deleting a corresponding object (i.e., state 1116); and (4) a column has been deleted (i.e., state 1118) and if so, deleting an entry from a hash table (i.e., state 1120). Ng, col. 8, ll. 29-58; Fig. 11A.

ANALYSIS

Based on the record before us, we find error in the Examiner's obviousness rejection of claim 4 which calls for deleting
the hashtable when the database and hashtable are not identical. Ng discloses different types of deleting when the database and hashtable are not identical (e.g., when changes in database information have occurred). *See* FF 11. For example, when a column has been deleted to a database's logical structure, Ng deletes an entry from the hashtable (*see id.*), but not the hashtable as required by claim 1. Ng describes another example where, when a table has been deleted from a database's logical structure, the entire object, including its hashtable, is deleted. *See id.* But in this situation, Ng does not save the database as a new hashtable as required by claim 1. *See id.* Although Ng may create or save a new object, provide a hashtable, or add or

save the hashtable with a new entry (*see id.*), Ng simply fails to delete a hashtable when the database and hashtable are not identical as required by claim 1.

For the foregoing reasons, Appellants have persuaded us of error in the obviousness rejection of: (1) claim 4 and (2) claims 14, 23, and 33 which recite commensurate limitations.

Claims 5, 15, 24, and 34

Regarding representative claim 5, the Examiner finds that Ng teaches the computer system linked to a network, and thus suggests storing information in a server or the computing device's virtual root. Ans. 6, 26. Appellants assert that Ng is silent about storing a new XML schema in a web server's virtual root and that the Examiner has not met the burden to demonstrate such storing is inherent in Ng. Br. 14. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 5 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested storing a XML schema in a web server's virtual root?

ADDITIONAL FINDINGS OF FACT (FF)

13. Ng's computer 101 is connected to the Internet 102. Ng's database 118 and source code 120 may be stored on other devices over the Internet 102. Ng, col. 4, ll. 55-56; col. 5, ll. 16-19; Fig. 1.

ANALYSIS

Based on the record before us, we find no error in the Examiner's obviousness rejection of claim 5 which calls for, in pertinent part, storing a XML schema in a web server's virtual root. As explained in the Answer's Response to Arguments (Ans. 26), the Examiner does not rely on Ng to teach an inherent feature of Ng. We therefore need not address whether the Examiner has demonstrated that Ng inherently teaches the limitations of claim 5. Br. 14. Also, as stated above, Ng and Srivastava collectively teach and suggest an XML schema and thus storing an XML schema within memory. *See* FF 3-9.

Ng further teaches that the computing device is connected to the Internet. *See* FF 3, 13. For this reason, we find the Examiner's position that Ng's disclosed computing device (e.g., 101) can act as web server reasonable, and thus, the computing device acts as a web server that stores the recited schema. Additionally, Ng states the source code (e.g., a schema, as explained above) can be stored on other devices in the network. *See* FF 13. Ng therefore also suggests storing the recited schema in locations other than the computing device, including a web server on the Internet's network.

Regarding storing the new XML schema in web server's "virtual root," a "root" is customarily defined as "[t]he main or uppermost level in a hierarchically organized set of information."[3] Employing the inferences and creative steps of an ordinarily skilled artisan, the ordinarily skilled artisan would have recognized storing data, including a XML schema, in the main or uppermost level of a server so that the rules for validating data within a computer file will be readily accessible to Ng's network users within a

[3] *Microsoft® Computer Dictionary* 574 (5th ed. 2002).

virtual environment. *See KSR*, 550 U.S. at 418. In any event, selecting a particular storage location within a hierarchy (e.g., the highest level or a lower level) for such information in a virtual system or otherwise would have been well within the level of ordinary skill in the art.

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of: claim 5 and claims 15, 24, and 34 not separately argued with particularity (Br. 14).

Claims 6, 16, 25, and 35

Regarding representative claim 6, the Examiner finds that that Ng discloses a hashtable and comparing the hashtable to the database. Ans. 3-4. Appellants state Ng's hashtable is not the same as Appellants' defined hashtable. Br. 15. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 6 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested a hashtable?

ADDITIONAL FINDINGS OF FACT (FF)

14. Appellants define a "hashtable" as "a computer file or memory allocation for storing a copy of a database, and in which the computer file or memory allocation is later compared to the database." Spec. ¶ 0019.

ANALYSIS

Based on the record before us, we find no error in the Examiner's obviousness rejection of claim 6 which calls for, in pertinent part, a hashtable. Appellants define a "hashtable" to include "a computer file or memory allocation for storing a copy of a database." FF 14. As the Examiner states (Ans. 3, 26), Ng discloses a hashtable (e.g., 310, 320) that is part of an object (e.g., 302, 304) of a data structure (e.g., 115). FF 5. Additionally, Ng's data structure's hashtables contain representative data found in a database (e.g., 118). For example, database 118 contains: (1) customer ID column 206 which corresponds to column object 312 (e.g., labelled customer ID); (2) name column 208 which corresponds to name object 314 (e.g., labelled name); and (3) order ID column 208 which corresponds to column object 322 (e.g., labelled order ID). *See* FF 4-5.

Ng further discloses that the object model's hashtables (e.g., 404, 412) have corresponding entries to the database tables' fields, including customer ID column 206 corresponds to entry 406 (labelled customer ID) and order ID column 212 corresponds to data entry 414 (labelled order ID). *See* FF 4, 6A. These disclosed hashtables thus also contain representative data from the database or store a copy of the database. Moreover, the data structure 115 and object model 116 containing these hashtables are stored in memory (e.g., 104) or memory allocation. *See* FF 3, 5, 6. These disclosed hashtables are therefore a memory allocation for storing a copy of a database as required by the Appellants-defined "hashtable" (*see* FF 14).

Regarding whether Ng's hashtable also satisfies the Appellants' defined "memory allocation is later compared to the database," Ng also teaches the comparing the hashtable to database data structures (FF 8),

which reflect the database's logical structure (*see* FF 3, 5, 6A). Because the database logical structure is a part of the database and because the database data structure reflects this structure, Ng's hashtable and its comparison steps also meet the Appellants' defined "hashtable" that require the memory allocation to be compared to the database.

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of: claim 6 and claims 16, 25, and 35 not separately argued with particularity (Br. 14-15).

Claims 7, 17, 26, and 36

Regarding representative claim 7, the Examiner finds that Ng teaches the recited features. Ans. 6. Similar to claim 6, Appellants argue that Ng's hashtable is not the same as Appellants' defined hashtable. Br. 15. We disagree for the reasons stated above in connection with claim 6.

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of: claim 7 and claims 17, 26, and 36 not separately argued with particularity (Br. 15).

Claims 8, 18, 27, and 37

Regarding claim 8, the Examiner finds that that Srivastava teaches notifying a registered party of an updated to the XML schema. Ans. 6-7. Appellants state Srivastava is silent about a registered party and updating the XML schema. Br. 15. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 8 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested notifying a registered party of an update to the XML schema?

ADDITIONAL FINDINGS OF FACT

15. Srivastava teaches a user or service provider can send a message to the service engine after an update is completed or can post a new location specification from which further updates will be provided. Srivastava, ¶ 0068.

16. Srivastava discusses notification services that can notify an administrator or other relevant personnel of certain system-generated events, such as a service execution failure. Trigger services also allow administrators to execute administrator-defined services upon the occurrence of an event. Srivastava, ¶¶ 0447-48.

ANALYSIS

Based on the record before us, we find error in the Examiner's obviousness rejection of claim 8 which calls for, in pertinent part, notifying a registered party of an update to the XML schema. Srivastava discloses an update service that allows a user (1) to send a message to service engine after an update is completed or (2) to post a new location specification from which further updates will be provided. FF 15. This teaching allows the user to notify a service engine of a completed updated but not a registered

party. *See id.* Also, Srivastava posts a location specification where updates are provided but this does not teach the posting's location specification is used to notify a registered party. *See id.*

Additionally, the Examiner cites to Srivastava's discussion of notifying an administrator or other relevant personnel of certain system-generated events. FF 16. Arguably, a system update could be a system-generated event. But we fail to see a motivation, absent impermissible hindsight, why an ordinarily skilled artisan would combine this general teaching with Ng's discussion of creating and updating an XML schema. The Examiner's reasoning for combining Srivastava's teaching with Ng is to notify users of changes and to provide an indication of modification. Ans. 7. Yet, the described indication in Ng relates to isolating database changes. *See* FF 7. At best, Ng's process notifies the computer that there is change and thus performs an update. *See id.* But Ng does not suggest notifying a registered party of this change in the XML schema.

For the foregoing reasons, Appellants have persuaded us of error in the obviousness rejection of: (1) claim 8 and claims 18, 27, and 37 which recite commensurate limitations.

Claim 9, 19, 28, and 38

Regarding claim 9, the Examiner finds that Srivastava teaches a database trigger to indicate a change in the database. Ans. 7. Appellants admit Srivastava disclose trigger services, but assert Srivastava is silent about a database trigger and using the trigger to indicate a change in the database. Br. 15. The issue before us, then, is as follows:

ISSUE

Under § 103, has the Examiner erred in rejecting claim 9 by finding that Ng, Janzig, and Srivastava collectively would have taught or suggested a database trigger to indicate a change in the database?

ANALYSIS

Based on the record before us, we find no error in the Examiner's obviousness rejection of claim 9. First, this obviousness rejection is not based on Srivastava alone. Thus, attacking Srivastava individually does not demonstrate nonobviousness when the rejection is based on combinations of references. *See Merck*, 800 F.2d at 1097. Second, Ng discloses a step (e.g., 506) that isolates changes to the database or a database mechanism (e.g., a trigger) that indicates a change in the database. *See* FF 7. Third, Srivastava further describes this concept by teaching providing a trigger service when an event occurs. *See* FF 16. Combining these teachings does no more than predictably yield a database trigger that indicates a change in the database (e.g., when changes are isolated) to update the object model and source code (*see* FF 5-8). *See KSR*, 550 U.S. at 416. We therefore disagree that Ng and Srivastava, when combined, are silent regarding a database trigger to indicate a change in a database as recited in claim 9.

Claims 19, 28, and 38 have not been separately argued (Br. 10-15) but are similar in scope to claim 9. We therefore group claims 19, 28, and 38 with claim 9 and will sustain these claims for similar reasons. *See* 37 C.F.R. § 41.37(c)(1)(vii).

For the foregoing reasons, Appellants have not persuaded us of error in the obviousness rejection of claim 9.

Claim 11

Regarding claim 11, the Examiner finds that that Ng creates an XML schema for a database in the first method. Ans. 9. Appellants refer to the arguments presented for claim 1. Br. 15. We are not persuaded for the reasons articulated above in connection with claim 1.

CONCLUSION

Under § 103, the Examiner did not err in rejecting claims 1, 3, 5-7, 9-11, 13, 15-17, 19, 20, 22, 24-26, 28, 29, 32, 34-36, and 38, but erred in rejecting claims 4, 8, 14, 18, 23, 27, 33, and 37.

ORDER

The Examiner's decision rejecting claims 1, 3-11, 13-20, 22-29, and 32-38 is affirmed-in-part.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a)(1)(iv).

AFFIRMED-IN-PART

pgc

DUKE W. YEE
YEE AND ASSOCIATES, P.C.
P.O. BOX 802333
DALLAS, TX 75380

EVIDENCE APPENDIX

Microsoft® Computer Dictionary 574 (5th ed. 2002).

Auto-Fill | Add A Page | Delete Last Page

Notice of References Cited	Application/Control No. 10/674,974	Applicant(s)/Patent Under Reexamination	
	Examiner Robert Timblin	Art Unit 2100	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
X	U	Microsoft® Computer Dictionary 574 (5th ed. 2002).
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Safari Books Online

ProQuest

Start a New Search | Entire Site | SEARCH | ADVANCED SEARCH

Sign Out & Clear Session · Personal Sign In

Microsoft® Computer Dictionary, Fifth Edition

By: Microsoft Press
Publisher: Microsoft Press
Pub. Date: May 1, 2002
Print ISBN-13: 978-0-7356-1495-6
Print ISBN-10: 0-7356-1495-4
Pages in Print Edition: 656
Amazon.com® Rating: (11 Ratings) Amazon.com® Reviews
Subscriber Rating: (0 Ratings)

START READING

BUY FROM PUBLISHER

Safari Books Online Subscribers Get Up to 35% OFF

Table of Contents

Copyright

PUBLISHED BY

Microsoft Press

A Division of Microsoft Corporation

One Microsoft Way

Redmond, Washington 98052-6399

Library of Congress Cataloging-in-Publication Data
Microsoft Computer Dictionary.--5th ed.
p. ; cm.
ISBN 0-7356-1495-4
1. Computers--Dictionaries. 2. Microcomputers--Dictionaries.

AQ76.5. M52267 2002
004'.03--dc21 200219714

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QWT 7 6 5 4 3 2

Distributed in Canada by Penguin Books Canada Limited.

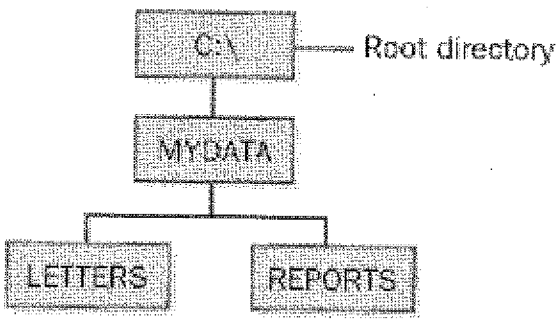
A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office or contact Microsoft Press International directly at fax (425) 936-7329. Visit our Web site at www.microsoft.com/mspress. Send comments to *<mspinput@microsoft.com>*.

root *n.* The main or uppermost level in a hierarchically organized set of information. The root is the point from which subsets branch in a logical sequence that moves from a broad focus to narrower perspectives. *See also* leaf, tree.

root account *n.* On UNIX systems, the account having control over the operation of a computer. The system administrator uses this account for system maintenance. *Also called:* superuser. *See also* system administrator.

root directory *n.* The point of entry into the directory tree in a disk-based hierarchical directory structure. Branching from this root are various directories and subdirectories, each of which can contain one or more files and subdirectories of its own. For example, in the MS-DOS operating system the root directory is identified by a name consisting of a single backslash character (\). Beneath the root are other directories, which may contain further directories, and so on. See the illustration.



Root directory.

root folder *n.* The folder on a drive from which all other folders branch. The root folder's name consists of a single backslash character (\). For example, on drive C, this folder would be represented in the file system as C:\.

rootless *n.* A mode in which an application belonging to a different user interface can run on top of a computer's underlying operating system without affecting that desktop or applications it may be running. For example, programs belonging to a rootless version of the X Window System can be run on a Mac OS X computer without disturbing the Aqua desktop. *See also* Mac OS X, X Window System.

root name *n.* In MS-DOS and Windows, the first part of a filename. In MS-DOS and earlier versions of Windows, the maximum length of the root name was eight characters; in Windows NT and later versions of Windows, the root name may be as long as 255 characters. *See also* 8.3, extension (definition 1), filename, long filenames.

root name server *n.* *See* root server.

root server *n.* A computer with the ability to locate DNS servers containing information about top-level Internet domains, such as com, org, uk, it, jp, and other country domains, in the Internet's Domain Name System (DNS) hierarchy. Beginning with the root server and continuing through referrals to name servers at lower levels of the hierarchy, the DNS is able to match a "friendly" Internet address, such as microsoft.com, with its numerical counterpart, the IP address. Root servers thus contain the data needed for referrals to name servers at the highest level of the hierarchy. There are 13 root servers in the world, located in the United States, the United Kingdom, Sweden, and Japan. *Also called:* root name server. *See also* DNS (definition 1), DNS server, top-level domain.

root web *n.* The default, top-level web provided by a Web server. To access the root web, you supply the URL of the server without specifying a page name or subweb.